


A Robust Dot-focused Classification Approach to Convolutional Braille Recognition


Wicus J. van der Linden

(Stellenbosch University, Stellenbosch, South Africa)

 <https://orcid.org/0009-0007-6203-0132>, 23263261@sun.ac.za)


Trienko L. Grobler

(Stellenbosch University, Stellenbosch, South Africa)

 <https://orcid.org/0000-0001-5274-0105>, tlgrobler@sun.ac.za)

Lynette van Zijl

(Stellenbosch University, Stellenbosch, South Africa)

 <https://orcid.org/0000-0001-5735-0448>, lvzijl@sun.ac.za)

Abstract: The effect of imbalanced data on the optical character recognition of Braille text is investigated by applying two techniques to a set of convolutional neural network image classification models. A multilabel classification framework is applied to identify the combination of Braille dots present in a character sample. This approach is compared to the multiclass classification framework prevalent in the literature, which directly identifies each sample as one of 64 possible Braille characters. Furthermore, data resampling methods are applied to investigate the impact of class imbalance on the multilabel and multiclass modelling approaches, respectively. The multilabel models are shown to achieve statistically significantly better performance than multiclass models, across different data resampling strategies. This includes better generalisation to out of distribution testing data from different Braille language codes, as well as robust performance under experimental image augmentation conditions. Furthermore, while multiclass models achieve better performance when trained on resampled data compared to training without resampling, this performance increase fails to rival the performance of the multilabel classification models across metrics and resampling strategies.

Keywords: Braille, Convolutional neural networks, Image augmentation, Image classification, Imbalanced Data, Multilabel classification, Optical character recognition, Resampling methods

Categories: I.4.8, I.5.1, I.5.2, I.5.4, I.7.5

DOI: 10.3897/jucs.161636

1 Introduction

Braille is a tactile reading and writing system used by the visually impaired community. To facilitate communication between the visually impaired and seeing communities, tools are needed to translate between Braille documents and documents written in natural language texts [Hsu, 2020]. Optical Braille recognition (OBR) systems are computer systems that allow for the scanning and subsequent transcription of Braille text [Calders et al., 1986].

In the last decade, OBR research has expanded to include the use of deep learning models, such as Convolutional Neural Networks (CNN) [Shimomura et al., 2018], with

various established CNN architectures applied to the recognition of Braille text [Elaraby et al., 2024]. Research is primarily focused on single-language texts, with few attempts to develop systems capable of multi-lingual Braille recognition [Al-Salman and Al-Salman, 2024]. However, datasets limited to Braille documents from a single language are subject to the character frequencies and imbalances present in that language. The effect of this imbalance on the subsequent OBR systems is sparsely researched.

The research objective of this study is twofold. Firstly, two classification methodologies are compared. The current standard multiclass classification approach directly identifies each sample as one of 64 possible Braille characters. This is compared to the novel multilabel classification approach introduced in this work, which instead identifies which of the individual Braille dots are present in a character. Secondly, the impact of class imbalance on both approaches is investigated, utilising data resampling techniques. This includes undersampling by removing training samples from the majority classes, and oversampling by synthesizing new samples for minority classes. This paper aims to show that the multilabel approach, in combination with data resampling, is less sensitive to imbalance in the data and yields recognition models that generalise to any language. This allows the development of robust, multi-lingual OBR systems without incurring additional dataset collection or model retraining costs.

Section 2 discusses relevant research on OBR and expands on its limitations and research gaps. The research methodology and experimental setup are described in Section 3. The results are presented and discussed in Section 4, and concluding remarks are provided in Section 5.

The processed Braille datasets, as well as all performance evaluations data used in this paper, are deposited on Zenodo at <https://doi.org/10.5281/zenodo.17453802>, and code samples and other assets are made available at <https://github.com/WvdL12/MultilabelBrailleOCR>.

2 Related Work

The first recognised OBR system, by Calders *et al.* in 1986, utilised algorithmic image processing techniques to locate and classify Braille dots and characters [Calders et al., 1986]. The use of neural networks by Morgavi *et al.* in 2002 resulted in a paradigm shift and the adoption of hybrid systems combining traditional and machine learning techniques [Morgavi and Mauro, 2002]. The use of CNN models for computer vision and object detection has increased rapidly since the publication of AlexNet in 2012 [Krizhevsky et al., 2012], and has been the focus of OBR research since 2018 [Shimomura et al., 2018].

The remainder of this section provides a summary of this research. Section 2.1 provides a brief overview of OBR system design, whereas Section 2.2 expands on the key techniques used in convolutional Braille recognition. Section 2.3 discusses dataset availability and preprocessing techniques. Section 2.4 concludes with the obstacles and limitations of the current research, identifying the research gaps present in the field.

2.1 Overview of Optical Braille Recognition

Braille recognition systems rely on the standardised structured nature of Braille characters. Characters are represented by cells of six dots arranged in three rows of two, where each dot can be raised (embossed) or flat. Altogether, these six dots form 64 unique combinations, or characters, including the empty character with no raised dots. This

design of Braille characters, including the size and spacing of Braille dots as seen in Figure 1a [Isayed and Tahboub, 2015], is standardised internationally. However, Braille codes exist for more than 130 different languages, each dictating the transcription between Braille and the corresponding natural language text [Devi and Baboo, 2012]. As with natural language alphabets, each Braille code uses the available characters in different combinations and frequencies.

Original OBR systems aimed to transcribe each detected Braille character into a corresponding natural language character, or the corresponding Braille ASCII character [Calders et al., 1986, Mennens et al., 1994]. Characters can be mapped to combinations of two or more Braille character cells, instead of a single cell, to allow more than 64 distinct combinations [Shao et al., 2022]. To accommodate this, OBR systems shifted to using abstracted labels that describe the dot-combination present in the cell [Isayed and Tahboub, 2015]. Two of the encoding formats used in literature include:

- Binary encoding uses a string of 6 bits, where bit i is set to 1 if the dot in position i is raised, and 0 otherwise.
- Decimal encoding interprets the binary encoded bit-string as a binary number, and converts it to the corresponding decimal number in the range $[0, 63]$.

The indexing order proposed in [Mennens et al., 1994] uses indices 1 through 6 as shown in Figure 1b, and has been accepted as the standard index order. The binary encoding lists the dot labels in this order, and dot position 6 is used as the most significant bit for decimal conversion [Al-Salman et al., 2007].

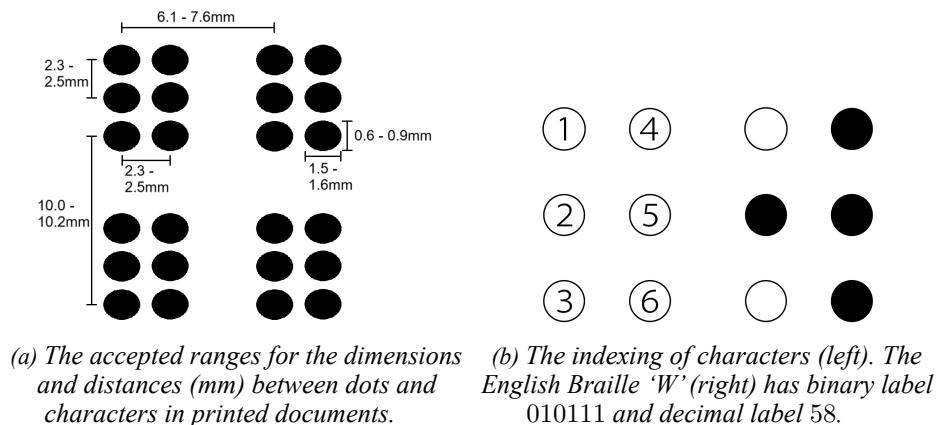


Figure 1: The standard structure and labelling of Braille characters [Isayed and Tahboub, 2015].

It is worth noting that the binary encoding format allows for an intuitive application of error correction methods common in coding theory. This can be combined with post-processing methods that detect errors based on spelling or grammar rules, to find the most likely correct characters based on the bit-differences between the binary labels [Antonacopoulos and Bridson, 2004, Shimomura et al., 2018].

2.2 Convolutional Braille Recognition

Deep learning CNN models are applied to OBR in two ways. Convolutional Braille detection systems utilise object detection or image segmentation frameworks to detect and extract Braille characters, and convolutional Braille classification uses classification models to classify isolated characters [Shimomura et al., 2018, Elaraby et al., 2024]. These systems can be chained together or otherwise combined to form a convolutional Braille recognition system for transcription of Braille documents [Baumgärtner et al., 2020]. Most research in the field is focused on classification of the isolated characters, while some studies have utilised single-pass detection frameworks to both extract and classify characters [Ovodov, 2021a, Bipin Nair et al., 2025].

Various established neural network architectures have been applied to both tasks, often with little to no modifications to the design. A list of these architectures are presented in Table 1. Some research also use these architectures as a feature extractor backbone, followed by a custom design classification head [Kausar et al., 2021, Meng et al., 2024], or as base models in an ensemble [Elaraby et al., 2024]. The best model design differs between applications, with both deep and shallow architectures achieving high classification performance [Kausar et al., 2021, Elaraby et al., 2024, Al-Salman and Al-Salman, 2024].

Custom classification models have also been designed and refined, using pairs of convolutional layers and pooling layers to extract features, and one or more dense layers to perform classification [Gezahegn et al., 2019, Revelli et al., 2022]. Hyperparameters such as kernel size, activation functions, learning rate and dense layer size are tuned to refine the model, achieving performance comparable to the applications using established architectures [Baumgärtner et al., 2020].

2.3 Braille data quality and availability

The quality of Braille documents vary depending on the lighting conditions, page degradation, camera quality and other external factors. The application of CNN models to Braille recognition yield systems that are flexible and robust to variability in these input quality and conditions [Hsu, 2020]. Data preprocessing is typically applied to simulate data quality conditions not captured in the training set, to further improve model generalisation to hold-out or test sets [Gonçalves et al., 2020, Kausar et al., 2021]. These augmentations include rotation, random noise, blur filters, and changing image contrast, brightness or colour values [Hsu, 2020].

Braille documents are also printed in two formats. While simple Braille documents are printed with raised dots on only one side of the page, many Braille documents have raised dots on both sides to reduce printing costs, a format known as interpoint Braille.

The difference between these formats can be seen in Figure 2. The valleys formed by the dots on the back of a page (called verso dots) are barely noticeable with tactile reading, but can only be visually distinguished from the dots on the front (recto dots) through the different shadow profiles [Isayed and Tahboub, 2015]. Shadows are found *below* the raised recto dots or *above* the valleys of verso dots, relative to the light source. Data quality factors obscure these patterns and increase the complexity of interpoint samples [Hanumanthappa and Murthy, 2016].

Another obstacle faced in research is the availability of data. Most research is performed on self-collected, proprietary datasets, leading to two key challenges in the field [Gezahegn et al., 2019, Shao et al., 2022].

Architecture	Seminal Paper	Applications
ResNet	[He et al., 2016]	[Li and Yan, 2021, Kausar et al., 2021, Al-Salman and Al-Salman, 2024, Elaraby et al., 2024, Alufaisan et al., 2021, Meng et al., 2024]
VGG	[Simonyan and Zisserman, 2014]	[Kausar et al., 2021, Murthy and Hanumanthappa, 2022, Al-Salman and Al-Salman, 2024, Elaraby et al., 2024]
GoogleNet	[Szegedy et al., 2015a]	[Shokat et al., 2020, Kausar et al., 2021, Elaraby et al., 2024]
AlexNet	[Krizhevsky et al., 2012]	[Kaur et al., 2020, Elaraby et al., 2024]
InceptionV3	[Szegedy et al., 2015b]	[Kausar et al., 2021, Elaraby et al., 2024]
DenseNet	[Huang et al., 2017]	[Kausar et al., 2021, Elaraby et al., 2024]
LeNet-5	[Lecun et al., 1998]	[Al-Salman and Al-Salman, 2024]
DarkNet	[Redmon and Farhadi, 2017]	[Elaraby et al., 2024]
SqueezeNet	[Iandola et al., 2016]	[Elaraby et al., 2024]
MobileNet	[Howard et al., 2017]	[Kumar and Basha, 2025]
RetinaNet	[Lin et al., 2017]	[Ovodov, 2021a, Shao et al., 2022]
YOLOv2/v5	[Redmon and Farhadi, 2017, Jocher et al., 2022]	[Gonçalves et al., 2020, Bipin Nair et al., 2025]

Table 1: List of CNN architectures commonly applied to convolutional Braille classification.

The first challenge is the labelling cost incurred to collect a large enough sample size to train high performing CNN models. Each sample needs to be digitally captured and labelled, with individual characters extracted from pages or provided with bounding box labels. Doing so manually is costly in terms of time and resources [Ovodov, 2021b], and a number of techniques have been utilised to alleviate or avoid these costs. Most notably, one approach uses a form of data resampling known as oversampling, where the available samples in a labelled dataset are duplicated to build up a larger training set. The augmentation techniques discussed above can be used to introduce variance among the resampled samples in the training data [Baumgärtner et al., 2020, Gonçalves et al., 2020, Li and Yan, 2021].

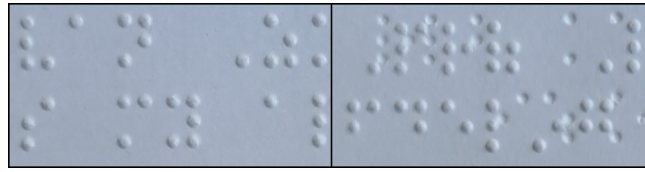


Figure 2: Extracts from a one-sided Braille page (left), and a two-sided (interpoint) Braille page (right). The light source is placed to the top right of each page, casting shadows to the bottom left of recto dots and on the top right side of verso dots.

The second challenge is limited performance comparisons between different techniques or models, due to both a lack of standardised datasets for comparison, and differing methods for evaluating performance on different techniques.

Most of the available datasets consist of isolated, cropped Braille characters, typically not covering all possible character codes. Examples include the Braille37 dataset produced in [Gezahegn et al., 2019], containing 37 distinct character codes, the Arabic Braille Character (ABC) set collected by [Elaraby et al., 2024] covering 32 distinct characters, and the Braille Character Dataset (BCD) containing 26 English Braille characters, available anonymously on Kaggle [Braille Character dataset, 2019]. Many of these datasets utilise augmented oversampling to increase the sample size for the included classes.

The Angelina [Ovodov, 2021a] and Double-Sided Braille Image (DSBI) [Li et al., 2018] datasets contain full document samples, with bounding box and character labels to facilitate training both Braille detection and classification models. These datasets contain large sample sizes and varying data quality conditions, without utilising augmentation or resampling. The Angelina and DSBI sets contain samples of 62 and 63 distinct Braille characters, respectively.

Despite the increased availability of datasets in recent years, Figure 3 shows a lack of adoption of these datasets, and the persistent use of proprietary datasets. Comparing reported results achieved on different datasets can yield misleading conclusions, and researchers often prefer to reproduce existing systems to compare with novel techniques rather than relying on reported results [Baumgärtner et al., 2020]. This is further complicated by incompatible performance metrics. Most recent research publications focus on character classifier models, reporting the attained classification accuracy, precision, recall or F_1 scores [Elaraby et al., 2024, Al-Salman and Al-Salman, 2024]. Other publications make use of object detection frameworks to detect and classify Braille characters with one model, reporting only the combined performance [Li et al., 2020, Ovodov, 2021a]. A few studies have utilised these object detection models to only detect Braille dots on a page, instead using post-processing to group and classify dots into characters [Shimomura et al., 2018, Meng et al., 2024].

Thus, despite multiple studies on convolutional Braille recognition, the field still lacks a standard and accepted way to compare performance and establish state of the art techniques and models [Meng et al., 2024].

2.4 Limitations in current research

Modelling techniques and architectures have been extensively researched and applied to Braille recognition. However, key limitations still arise due to the difficulties discussed above.

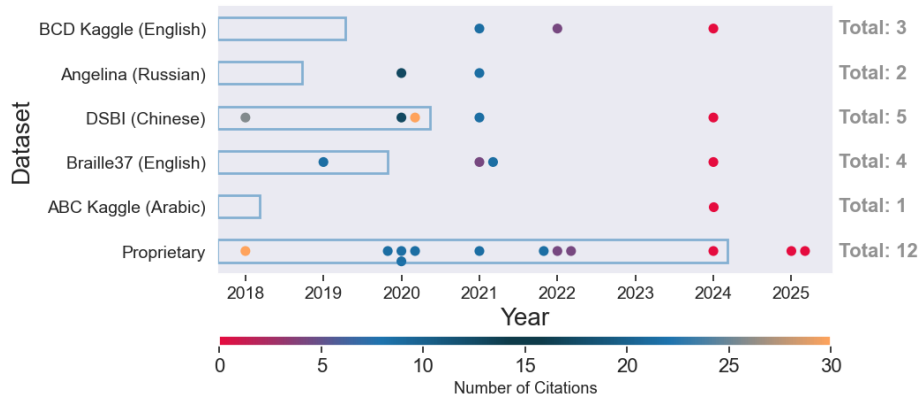


Figure 3: The use of different Braille datasets in convolutional systems since 2018. Each data point represents a publication, with its colour indicating the number of times it has been cited. The x and y-axes denote the publication year and the Braille dataset used, respectively. The total number of publications for each dataset is highlighted with horizontal bars. The persistent preference for using proprietary datasets can be seen both in the quantity and recency of publications.

As mentioned, different natural languages have different codes for transcribing between Braille and natural text, and use subsets of Braille character codes in varying frequencies [Devi and Baboo, 2012, Al-Salman and Al-Salman, 2024]. While some studies utilise these transcription rules character frequencies to improve accuracy through error detection and correction [Shimomura et al., 2018, Gonçalves et al., 2020, Shao et al., 2022], few studies have considered building a multilingual OBR systems [Sivasamy et al., 2013, Al-Salman and Al-Salman, 2024].

These studies simply utilise Braille-to-text lookup tables corresponding to each language, without addressing the limitations within the datasets used for training, limiting the system’s performance on languages not included in the study. In particular, most models are trained on only 26 to 37 distinct Braille character classes (see the datasets listed in Section 2.3). These models learn only the features and patterns of those characters, and are unable to recognise any characters not represented in the training data.

While datasets sourced from diverse Braille documents typically include more distinct characters, as with the DSBI and Angelina datasets, these datasets are still subject to the imbalances present in natural language. For example, the most common Russian Braille character occurs nearly 40 times more often than the least common character [Ovodov, 2021a]. The presence of such imbalance is known to cause complications in machine learning models, which often underperform on minority classes. These less common characters may be more common in other languages or Braille codes, which could lead to significantly poor performance when an OBR model trained on one language is tested on a different language, known as out-of-distribution (OOD) testing, incentivising researchers to produce different models for different languages.

One study shows this lack of generalisation explicitly, training a model on the Chinese DSBI dataset, and evaluating in-distribution (ID) on the DSBI set and OOD on the Russian Angelina dataset [Ovodov, 2021a]. In the ID evaluation, the model achieved an average class-wise F_1 score of 0.9976, similar to the classification performance of models trained

on completely different datasets reported by recent studies [Al-Salman and Al-Salman, 2024, Elaraby et al., 2024]. In the OOD evaluation, however, the model achieved a score of only 0.9265. The drop in performance is especially evident in the average class-wise recall, where the model achieved a score of 0.9975 ID, while only achieving a score of 0.8980 OOD.

2.5 The goal for this study

The impact of character imbalance on OBR model performance and generalisation has not been formally investigated. Established techniques to compensate for class imbalance, such as data augmentation and selective resampling, has not been applied to directly address Braille character class imbalance.

This study aims to address class imbalance by introducing the use of the multilabel classification framework to the task of Braille recognition. The binary encoding scheme used in classical OBR systems, is used to design a multilabel classifier that yields a binary prediction for each dot position, with the combination of these six predictions uniquely identifying the Braille character. It is hypothesised that the impact of character imbalance on models using this framework is mitigated, as predictions are made based on the absence or presence of each dot individually, rather than the exact combination of dots defining the Braille character.

To assess the impact of class imbalance, data resampling and augmentation is used to construct training sets with different class distributions. Due to the difficulties noted in comparing model performance across different publications and datasets, both multilabel and multiclass models are trained on each resampled dataset, with independent hyperparameter tuning. This facilitates a fair comparison of the two frameworks, without relying on the reported performance of state-of-the-art multiclass classification models developed in recent publications.

3 Methodology

The methodology of this study provides details on the data processing, model design, hyperparameter tuning, data resampling strategies, and performance evaluation.

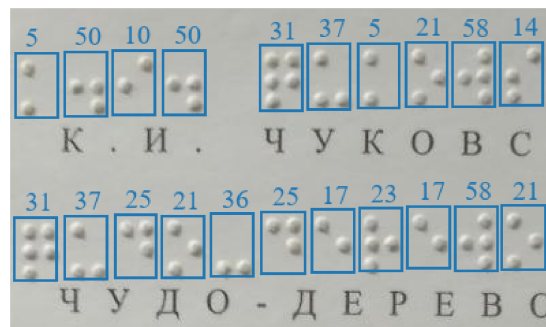
3.1 Data preprocessing

This study prioritised the use of diverse real-world Braille document data, with minimal use of preprocessing techniques like synthetic noise, rotation or other augmentations. The large sample size and extreme class imbalance of the Russian Angelina dataset [Ovodov, 2021a] is well suited to explore the impact of this imbalance on model performance. To explore the generalisation of OBR models to different languages, the Chinese DSBI dataset [Li et al., 2018] was used as an out-of-distribution (OOD) test set.

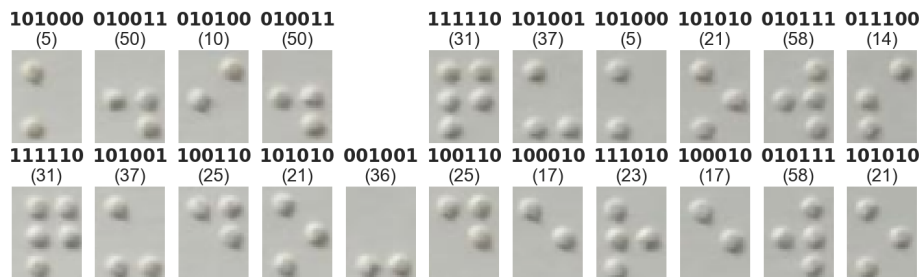
The Angelina dataset includes pages from different documents with different page and image qualities. A stratified split procedure was used to divide these pages into training, validation and testing sets, such that the distributions of document sources and Braille character classes were similar across each set. From each page, the individual Braille characters were extracted using the provided character boundary coordinates, to obtain sets of character samples with their corresponding target labels. The training set consisted of roughly 76% of the Braille character samples, and the validation and testing sets roughly 11% and 13% respectively.

The only preprocessing applied was resizing all extracted characters to a common size, maintaining three RGB colour bands and adhering to the standardised Braille cell aspect ratio of 3 : 4 [Isayed and Tahboub, 2015]. The target labels were standardised and stored in two formats: binary encoding labels for multilabel models, and a one-hot encoding obtained from the decimal encoding format for multiclass models. All pages from the documents in the DSBI dataset were combined into an OOD test set, with individual characters extracted and standardised by the same procedure as the Angelina dataset.

Figure 4a depicts an annotated sample of a Russian Braille page from the training set. The boundary coordinates and class labels of each character in the page are annotated in blue. These coordinates are used to extract each character along with its associated label, with results shown in Figure 4b.



(a) An annotated sample from a document containing both Russian and Braille text. The provided Braille character boundaries and decimal labels are annotated in blue.



(b) The corresponding extracted and standardised Braille character samples, resized to 30×40 pixels and titled with binary labels in bold and decimal labels in parentheses.

Figure 4: A showcase of the Braille character extraction results.

3.2 Model design and hyperparameters

All models developed in this study follow a simplified VGGNet architecture. The hidden layers consist of two convolutional blocks followed by a dense linear layer. Each

convolutional block includes two convolutional layers, followed by a max pooling layer. The details of each layer is determined by the hyperparameters listed below.

- The parameter F determined the number of convolutional filters per convolutional layer, with the first block of convolutional layers using F filters, and the second block using $2F$ filters. F could take on integer values between 8 and 64, inclusive.
- The size of all convolutional kernels was set to $K \times K$, where K is an integer between 2 and 5, inclusive. The pooling kernel sizes were fixed at 2×2 .
- A padding of zero-pixels could be added around each sample to prevent the loss of details in the border through the convolution operations. This padding was determined by a binary hyperparameter P , where a value of 0 indicated no padding, also known as *valid* padding, and 1 indicated a padding width of 1 pixel.
- The size of the dense layer included before the output layer was determined by D , an integer in the range [25, 150].

An activation function was applied after each convolutional layer, as well as the dense layer. These univariate functions transform the outputs of each neuron in the layer, introducing non-linearity and controlling the output ranges of each neuron. A hyperparameter was used to determine the function used across the hidden layers of each model, with the three functions listed here included as options.

The Rectified Linear Unit (ReLU) is a nonnegative function defined as

$$f(x) = \max(0, x). \quad (1)$$

The LeakyReLU variant of this function uses a small positive gradient α for negative inputs, defined piece-wise as

$$f(x) = \begin{cases} \alpha x & \text{if } x < 0, \\ x & \text{if } x \geq 0. \end{cases} \quad (2)$$

While the gradient could be tuned as another hyperparameter, for the purpose of this study, it was fixed at $\alpha = 0.01$.

Lastly, the tanh function uses the hyperbolic tangent function, defined in terms of Euler's number e as

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (3)$$

The design of the output layer of each model in the study depends only on the classification framework utilised. This design primarily dictates the number of output neurons and the activation function applied, but also determines the loss function used during training to measure and optimise the model's performance.

In the multiclass models, the output layer consists of one neuron for each of the 64 possible character classes¹, where each neuron output $\sigma_{i,j}$ for a given input sample i represents the predicted probability that sample i belongs to class j . The final classification by the model for a given sample is the class with the highest predicted

¹ The data used for this study does not explicitly include the empty Braille character, or class index 0. The output node for this class was included to be compatible with the decimal encoding, and for consistency with studies that do include samples for the empty character.

probability, corresponding to a decimal encoding label in the range $[0, 63]$. To ensure that the output of each neuron satisfies a valid probability range of $(0, 1)$, and to ensure that the sum of all outputs equates to 1, the softmax activation function is applied, defined as

$$\sigma_{i,j} = \text{Softmax}(\mathbf{z}_i, j) = \frac{e^{z_{i,j}}}{\sum_{k=0}^{C-1} e^{z_{i,k}}} \quad (4)$$

where \mathbf{z}_i denotes the vector of outputs, or logits, for sample i , and $C = 64$ corresponds to all possible character classes, offset by one to align with the decimal encoded labels.

For multilabel models, the output layer consists of one neuron for each of the six Braille dot positions, ordered as indicated in Figure 1b. In this case, each neuron output $\sigma_{i,j}$ for a given input sample i represents the predicted probability that a dot is present at position j . The final classification by the model for a given sample is the binary encoded label obtained across these dot positions, taking predicted probabilities greater than or equal to 0.5 as a present dot (also called a *positive* prediction, denoted by 1) and probabilities less than 0.5 as absent dots (or *negative* prediction, denoted by 0). Each neuron only needs to satisfy the valid probability range of $(0, 1)$, independent of the other neurons, and as such the sigmoid activation function is applied, defined as

$$\sigma_{i,j} = \text{Sigmoid}(\mathbf{z}_i, j) = \frac{1}{1 + e^{-z_{i,j}}} \quad (5)$$

where \mathbf{z}_i denotes the logits for sample i .

The one-to-one correspondence of decimal and binary encodings allows for direct comparisons between model outputs, and equivalent evaluations of model performance, despite the different classification processes and output formats.

During model training, the predicted probabilities can be utilised to evaluate loss functions, that measure how well the model outputs match the ground truth labels of the input samples. Larger losses indicate more erroneous predictions, and prompt larger changes to the weights associated to the neurons within the model's hidden layers. An optimiser function determines how and where to apply these changes at each training step, by propagating the loss back through the network and adjusting neuron weights appropriately.

The categorical cross-entropy loss function is used in training multiclass models. For input sample i belonging to class c_i , let \mathbf{y}_i be a one-hot encoded target vector with $y_{i,c_i} = 1$ and zeroes elsewhere, and let $\hat{y}_{i,j} = \text{Softmax}(\mathbf{z}_i, j)$ be the model's predicted probability that sample i belongs to class j . Then the categorical cross-entropy loss over N samples is defined as

$$L_{\text{CCE}}(Y, \hat{Y}) = - \sum_{i=0}^{N-1} \sum_{k=0}^{C-1} y_{i,k} \log(\hat{y}_{i,k}) \quad (6)$$

$$= - \sum_{i=0}^{N-1} 1 \cdot \log(\hat{y}_{i,c_i}). \quad (7)$$

The binary cross-entropy loss function is used in training multilabel models. For input sample i with binary encoded target vector \mathbf{y}_i , let $\hat{y}_{i,j} = \text{Sigmoid}(\mathbf{z}_i, j)$ be the model's predicted probability that position j contains a Braille dot. Then the binary

cross-entropy loss over N samples, over the $d = 6$ positions², is defined as

$$L_{\text{BCE}}(Y, \hat{Y}) = - \sum_{i=0}^{N-1} \sum_{k=0}^{d-1} [y_{i,k} \log(\hat{y}_{i,k}) + (1 - y_{i,k}) \log(1 - \hat{y}_{i,k})]. \quad (8)$$

For both frameworks, the Adam optimiser [Kingma and Ba, 2014] was used for weight updates. The learning rates used in the optimiser are determined by three hyperparameters $lr \in (1 \times 10^{-4}, 2 \times 10^{-2})$, $\beta_1 \in (0.85, 0.95)$ and $\beta_2 \in (0.99, 0.9999)$. Additionally, weight decay, or l_2 regularisation, was applied to the model weights to mitigate model overfitting. The regularisation strength was determined by the hyperparameter $l_2 \in (1 \times 10^{-4}, 3 \times 10^{-2})$.

Figure 5 shows an example model based on this design, using hyperparameters $F = 16$, $K = 5$, $P = 1$ and $D = 100$. The number of output neurons corresponds to a multilabel target of six dot labels. The dimensions of layer i depends on the layer type, the dimensions of the previous layer, and the hyperparameters K and P . For a convolutional layer, the height and width are determined by Eq. 9 and Eq. 10 respectively. A max pooling layer replaces each (non-overlapping) kernel with its maximum valued pixel, reducing both dimensions by half.

$$h_i = h_{i-1} - K + 2P + 1 \quad (9)$$

$$w_i = w_{i-1} - K + 2P + 1 \quad (10)$$

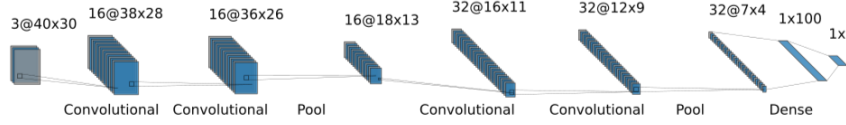


Figure 5: An example of the common CNN architecture used, with hyperparameters $F = 16$, $K = 5$, $P = 1$ and $D = 100$. The input layer contains the RGB-colour image, with height $h_0 = 40$ and width $w_0 = 30$.

3.3 Hyperparameter tuning

Hyperparameter tuning was applied using an extended iterated F-race [Birattari et al., 2002, Balaprakash et al., 2007, Klazar and Engelbrecht, 2014] to explore the search space for an optimised model. The F-race is a phased procedure, with each phase involving training and evaluating a set of model candidates. The training set was divided into 10 equal sized folds, and in each F-race phase a different set of nine folds were used as

² The indices 1 through 6 shown in Figure 1b are shifted to indices 0 through 5 to align with array indexing.

training data, while the remaining fold was used as validation data. Each model candidate was trained for up to 20 epochs, stopping early if the loss on the validation data did not improve for three consecutive epochs.

The minimal loss of each candidate on the validation data was recorded after each training phase, and a non-parametric Friedman test was applied to these loss scores. Candidates with a significantly high loss compared to other candidates were removed from the candidate set, and not included in further training and evaluation phases.

The iterated F-race was initiated by generating 20 model candidates, using a uniform random distribution for each hyperparameter space to sample parameters for each candidate. The F-race procedure was repeated until all 10 distinct fold-combinations are used, or until five or fewer candidates remain. One of the remaining candidates was selected randomly, with selection probabilities proportional to each candidate's mean validation loss across the F-race phases. The hyperparameters of this candidate was used to update the distribution of each hyperparameter space, increasing the probability density around the relevant parameter, and decreasing density elsewhere. A set of 19 model candidates are generated by sampling random hyperparameters from the updated distributions, and are used along with the selected candidate to form the set of 20 candidates for the next iteration.

This F-race procedure was repeated for five iterations with multiclass model candidates, and separately for five iterations with multilabel model candidates. In each case, the model candidate with the lowest mean validation loss across iterations was selected as the optimised model configuration. The optimised multiclass and multilabel model configurations were trained on the full training set for up to 100 epochs, using the original validation set to stop training early if validation loss did not improve for 3 consecutive epochs.

3.4 Data resampling

Two forms of data resampling were used to investigate the impact of class imbalance on model performance. Oversampling aims to increase the sample size of minority classes in the training set. This typically includes selecting random training samples to duplicate, often adding augmentation to the duplicated samples to increase the variance in the oversampled set. Undersampling aims to reduce the sample size of majority classes, by randomly removing samples from the training set.

For the purpose of oversampling, rotation and reflection augmentations are applied to duplicates of selected training samples, to obtain new samples. In particular, by reflecting a Braille character in either axis, or rotating the character by 180° , the augmented character could belong to a different character class, with different Braille dot combinations, as shown in Figure 6.

The oversampling procedure then involves identifying which training samples to rotate or reflect, in order to obtain new samples of the desired minority classes. Additional restrictions are applied to ensure that each original training sample is duplicated at most three times, once with each augmentation type (rotation, and reflection in two directions).

In undersampling the majority classes, care was taken to avoid undersampling informative samples with large contributions to model training. The classification error rate of a model on each training sample can be used to distinguish between informative and noninformative samples. At each iteration, or epoch, of the model training process, the model's prediction for each training sample is recorded. After training, the error rate of a given training sample corresponds to the proportion of epochs where the model incorrectly classified the sample. Training samples with low error rates contribute less to

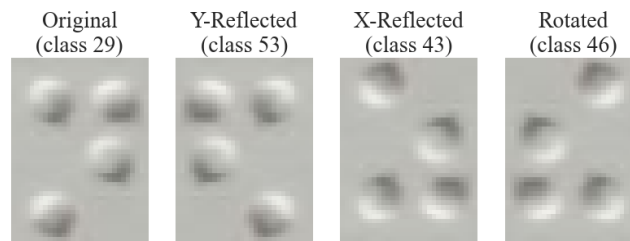


Figure 6: An example of augmentations applied to a majority class sample. Roughly 4.9% of the training samples belong to character class 29, while the augmented samples belong to the less frequent character classes 43, 46 and 53, each corresponding to between 1.5% and 1.86% of training samples.

the loss function during training, and therefore have less impact on the weight updates through backpropagation [Liu et al., 2009, Ke et al., 2017].

An error threshold is used to obtain an explicit distinction between informative and noninformative samples, with error rates below the threshold corresponding to noninformative samples. Using a higher threshold would permit more undersampling, and a greater reduction in class imbalance, while a lower threshold limits undersampling and maintains a larger training sample size. The aim of this study is primarily to establish the impact of resampling on model performance, and as such the threshold was selected to divide the training set approximately equally into informative and noninformative samples. Fine-tuning this threshold to find an optimal degree of undersampling is deferred to further study. The undersampling process then involves identifying noninformative training samples from majority classes to remove from the resampled training set.

This study investigated three resampling scenarios to obtain three different resampled sets for model training.

The *no resampling* (NR) scenario corresponds to a baseline result, in which no resampling was applied to the training data. Sample error rates are recorded while training the final multiclass and multilabel models in this scenario, with these error rates averaged to obtain a final error rate per training sample. A threshold of 0.075 was used to divide the training set into 55% informative samples and 45% noninformative samples, ensuring that no more than half of the training set could be undersampled.

In the *class resampled* (CR) scenario, resampling was applied to maximise the Shannon entropy as a measure of imbalance between Braille character classes³. Resampling was applied by undersampling noninformative samples from the largest character class, and identifying appropriate samples to oversample and augment to obtain new samples of the smallest character class. This process was repeated iteratively, with the largest and smallest classes changing as samples are removed and added, until the Shannon entropy reached a desired threshold, or until no candidates exist to oversample for the minority class or to undersample from the majority class.

In the *label resampled* (LR) scenario, resampling was applied to minimise the absolute binary correlation between the dot-position labels. Strong positive correlations arise due to the co-occurrence patterns of pairs of dot positions, where most training samples either

³ The Shannon entropy is the normalised form of the Shannon diversity index, or Shannon-Weiner index. It measures the degree of uncertainty when predicting the class of a randomly selected sample. The uncertainty is minimised when the population contains only a single class, and maximised when the population contains two or more equally sized classes.

have both or neither of these dots present. Strong negative correlations, which also lead to strong absolute correlations, arise between two dot positions when training samples often have exactly one of the two dots present.

Noninformative samples that include a strongly correlated dot-pair are undersampled, and samples are duplicated and augmented to obtain more instances of low-correlation dot-pairs. This is again repeated iteratively until the absolute correlation is below a desired threshold, or no appropriate over- or undersampling candidates are identified.

For each resampling scenario, independent hyperparameter tuning (see Section 3.3) was performed for the multiclass and multilabel modelling approaches. Resampling was applied to each training fold separately, to avoid data leakage in the validation folds. Within each resampling scenario, the same resampled training data were used in tuning both the multilabel and multiclass models, and the same resampled full training set was used to train the optimised models.

3.5 Performance metrics

Six key performance metrics were recorded for each trained and optimised model, including error measures and dot-centric recall, precision and F_1 metrics.

Error measures report the proportion of incorrect predictions across an evaluation set. Error can be measured with respect to character class predictions, or dot label predictions. For a given sample, a correct class prediction by a multiclass model corresponds to correct predictions for all six dot labels, while an incorrect class prediction necessitates at least one incorrect dot label prediction. As such, recoding multiclass class predictions from decimal encodings to the corresponding binary encodings, as discussed in Section 2.1, allow all metrics to be calculated with respect to the binary labels.

Consider an evaluation set of N samples, with target matrix $Y_{N \times 6}$ of binary labels, and prediction matrix $\hat{Y}_{N \times 6}$. Let $I(p)$ be an indicator function evaluating to one when the statement p is true, and zero otherwise. The product of multiple indicator functions evaluates to one only when each individual indicator evaluates to one.

Class error measures the proportion of incorrect character class predictions. Using indicator functions as described above, and incorrect class prediction is defined as

$$I(\text{Incorrect } \hat{Y}_i) = I_{\hat{Y}_i} = 1 - \prod_{d=1}^6 I(Y_{i,d} = \hat{Y}_{i,d}). \quad (11)$$

Then the overall class error can be formally calculated as

$$\text{Class error} = \frac{\sum_{i=1}^N I_{\hat{Y}_i}}{N}. \quad (12)$$

On the other hand, label error measures the proportion of incorrect dot label predictions, formally calculated as

$$\text{Label error} = \frac{\sum_{i=1}^N \sum_{d=1}^6 I(Y_{i,d} \neq \hat{Y}_{i,d})}{6N}. \quad (13)$$

To further investigate the relationship between class and label errors, a mean error metric is derived to measure the expected number of mislabelled Braille dots for each incorrect class prediction. This mean error metric corresponds to the bit-differences

between binary labels used by the error correction approaches in classical OBR systems [Antonacopoulos and Bridson, 2004], serving as an indication of the degree of character misclassification by each model.

This mean error metric takes a real value in the range $[1, 6]$, and can be formally calculated as

$$\text{Mean error} = \frac{\sum_{i=1}^N \sum_{d=1}^6 I(Y_{i,d} \neq \hat{Y}_{i,d})}{\sum_{i=1}^N I_{\hat{Y}_i}} \quad (14)$$

which is proportional to the ratio of label error to class error.

Recall, precision and F_1 scores allow for deeper analysis of a model's performance on the Braille dot level, providing further context to the error measures above. To understand the recall, precision and F_1 scores for dot label predictions, first consider again the interpretation of a given label prediction $\hat{Y}_{i,d}$.

A prediction of $\hat{Y}_{i,d} = 1$ corresponds to a *positive* prediction, and predicts that a dot is present at dot position d . A prediction of $\hat{Y}_{i,d} = 0$ corresponds to a *negative* prediction, and predicts that a dot is absent at dot position d . The total *true positive* (TP) and *true negative* (TN) predictions describe the number cases where the model correctly predicted the dot labels, while the total *false positive* (FP) and *false negative* (FN) predictions describe incorrect label predictions.

The (label) recall then measures the proportion of present Braille dots in the evaluation set that were correctly detected. That is

$$\text{Recall} = \frac{TP}{TP + FN}. \quad (15)$$

The (label) precision, on the other hand, measures the proportion of a model's *positive* predictions that are correct. That is

$$\text{Precision} = \frac{TP}{TP + FP}. \quad (16)$$

The (label) F_1 score is defined as the harmonic mean of *recall* and *precision*, formally calculated as

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (17)$$

$$= \frac{2TP}{2TP + FP + FN}. \quad (18)$$

3.6 Performance evaluation

Each of the three resampling scenarios produced an optimised multiclass model, and an optimised multilabel model, resulting in six independently optimised and trained models.

Overall performance was evaluated in-distribution (ID) on the Angelina test set, and out-of-distribution (OOD) on the DSBI dataset.

Each model's robustness to varying data quality and conditions was investigated by applying randomised augmentation trials on the ID evaluation set. The experimental methodology is discussed further in Section 3.6.1.

Lastly, the impact of class imbalance on performance was investigated by evaluating model performance on groups of Braille character classes. Section 3.6.2 provides more details on these groups.

3.6.1 Experimental methodology

Three additional performance evaluations were conducted on augmented variations of the ID evaluation set. In each experiment, the augmentations were applied randomly to each test sample X_i of shape $(3, 40, 30)$, consisting of standardised pixels $p_j \in [0, 1]$.

The brightness experiment applied randomised pixel brightness reduction to simulate poor lighting conditions. For each test sample, a random variable $\gamma_i \in [1.0, 1.3]$ was sampled independently, with each pixel value in the sample replaced with a reduced pixel value $p'_j = p_j^{\gamma_i}$. The cases with $\gamma_i = 1.0$, corresponding to no pixel change, served as experimental control.

In the noise experiment, Gaussian white noise was added to each sample to simulate distortion or other forms of noise in digital images. For each test sample, a random variable $\sigma_i \in [0.0, 0.1]$ was sampled independently, used to sample white noise from a multidimensional Gaussian distribution with mean $\mu = 0$ and standard deviation σ_i . A standard deviation of 0 resulted in no added white noise, serving as experimental control.

Lastly, random image rotation was applied in the rotation experiment, with each test sample rotated by a random angle $\theta_i \in [-25^\circ, 25^\circ]$. Cases with no rotation, corresponding to $\theta_i = 0^\circ$, are included as experimental control.

Each experiment was repeated 15 times, and for each trial, the augmentation parameters (γ_i , θ_i and σ_i) were sampled independently for each test set sample, from a discretised sub-domain. The results across 15 trials were aggregated to obtain means and standard deviations of each performance measure.

Model performance was evaluated for the entire augmented test set, as well as on subsets of test samples grouped by sampled augmentation parameter. This allowed insight into the overall robustness of the model, as well as the relationship between augmentation intensity and model error.

3.6.2 Braille class groups

Three groups of character classes were identified, based on the character class frequencies in the unaltered training set — minority, majority, and middle classes. The performances per character class were calculated based on the model predictions on the OOD evaluation samples for each class. This yielded a performance measure per character class for each model, treating each class equally irrespective of the number of samples present.

The minority group includes classes that each account for less than 0.1% of the training data. This includes 17 character classes, each containing fewer than 70 training samples in the *no resampling* scenario. Most minority classes includes between 184 and 1723 OOD evaluation samples, each accounting for roughly 0.4 – 3.8% of the evaluation set. A notable exception is character class 16, which includes only 7 training samples (ranked 58th in the training set), but corresponds to the largest OOD character class, with 3639 evaluation samples (roughly 8% of the set).

The majority group includes classes that each account for more than 2% of the training data, including 17 character classes with between 1400 and 6200 samples. Each majority class includes between 239 and 1821 OOD samples, each accounting for 0.5 – 4.0% of the set.

The remaining 29 character classes each account for between 0.1% and 2% of the training data, and are grouped together as middle classes. This group is used as a performance baseline to investigate the majority and minority class performances. Most middle classes include between 84 and 1897 OOD samples, or roughly 0.2 – 4.2% of the set, with the exception of class 40 with only 2 OOD test samples.

4 Results

The overall model performance is discussed first, including the ID and OOD evaluations, as well as the overall performance evaluation of the three experiments. This is followed by an investigation of the relationship between augmentation intensity and model performance. Lastly, an analysis of the impact of class imbalance on performance is provided.

All performance metrics are reported as error rates — a precision, recall or F_1 score of x is inverted to an equivalent error score $1 - x$. This allows for consistency across metrics, such that in each a lower value indicates better performance.

4.1 Overall performance

The overall performance measures of the six models (see Section 3.6) on the five evaluation sets (ID, OOD, and three experiments) are included in Tables A1–A6 in Appendix A. In each table, the pairwise comparison between modelling approaches is emphasised by highlighting the performance of the best model within each resampling scenario, in bold. Furthermore, to aid in comparing different resampling strategies for a given modelling approach, the performance of the best multiclass and multilabel models in each evaluation, are underlined.

The pairwise comparison of multiclass and multilabel model performance on the ID and OOD evaluation sets reveals that multilabel models achieved better performance in nearly all metrics, across the different resampling scenarios. The same observation holds under the noise and brightness experiments. Under the rotation experiment, results vary more across different metrics, particularly in the *class resampled* scenario.

Overall, this pairwise comparison shows that multilabel models achieve better performance in all metrics. This is supported with statistical testing at a significance of $\alpha = 0.05$, using the performance results across all resampling scenarios and evaluation sets to apply a paired-sample test for each metric. The null hypothesis proposes no significant performance difference between the two modelling approaches. The one-sided alternative hypothesis proposes that the multilabel models yield better performance. A Shapiro-Wilk test was used to confirm that none of the performance sets match a normal distribution [Shapiro and Wilk, 1965], and as such the non-parametric Wilcoxon signed-rank test was used for analysis [Wilcoxon, 1945]. The null hypothesis is rejected in each metric, with the corresponding p-values indicated in Table 2.

Statistical analyses of the three resampling scenarios indicate that not enough evidence is present to reject the null hypothesis, which states that the different scenarios yield equivalent performance. However, an inspection of the effect of different resampling scenarios on each modelling approach shows that data resampling does yield improved performance for multiclass models. Multilabel models, on the other hand, show more varied performance across different evaluations different metrics, exhibiting no direct benefit from resampling. In most metrics, the best multiclass models are those trained on

resampled data — with the exceptions of the mean error on the OOD evaluation, as well as the precision in the experimental evaluations.

A rank-based analysis further supports this conclusion. For each evaluation set and each metric, the different resampling scenarios are assigned ranks 1 through 3, with rank 1 assigned to the best performing scenario, and rank 3 assigned to the worst. These performance ranks are averaged across the five evaluation sets, yielding a mean performance rank for each resampling scenario, per modelling approach and per metric. These performance ranks are reported for the two modelling approaches in Tables 3 and 4.

For multiclass models, the *no resampling* scenario achieves the worst average rank in most metrics, with the exception of mean error and precision. This indicates that multiclass models trained on resampled data tend to provide more positive Braille dot predictions, which would lead to lower precision and higher recall rates. One interpretation is that these models do not accurately differentiate recto- and verso-dots, as the dot shadow-patterns discussed in Section 2.3 are inverted in rotated and reflected Braille samples, causing ambiguity and noise in the training data. However, both *class resampled* and *label resampled* scenarios yield better average rankings in both accuracy metrics, as well as F_1 , indicating that the models still achieve better overall performance, despite a high positive label prediction rate. Although these results indicate that data resampling does yield performance improvements for multiclass models, it is difficult to conclusively state which resampling strategy yields the best improvement. The *class resampled* strategy achieves the best overall rank, with the overall rank of the *label resampled* strategy only slightly worse. *class resampled* outranks *label resampled* in recall and precision, while the latter achieves a better rank in class error, and the ranks are tied for label error, mean error and F_1 score.

Considering multilabel models, on the other hand, both scenarios that employ data resampling achieve poorer performance overall. They achieve better recall rates compared to the *no resampling* scenario, similar to multiclass models, and the *class resampled* model interestingly yields the same mean rank as *no resampling* in precision. However, *no resampling* still achieves the best mean rank in each of the remaining four metrics, as well as in the overall rank. This demonstrates that although data resampling mitigates class imbalance, it has a detrimental impact on multilabel model performance. This could align with the hypothesis that the multilabel approach is less susceptible to class imbalance, and thus has less to gain from mitigating this imbalance. The possible downsides of undersampling potentially useful training samples, and increasing the ambiguity between recto- and verso-dots, likely outweighs the potential benefits of improved class balance.

Performance Metric	Wilcoxon test p-value	Result
Class error	2.136×10^{-4}	Reject H_0
Label error	9.16×10^{-5}	Reject H_0
Mean error	3.05×10^{-5}	Reject H_0
Recall	1.0071×10^{-3}	Reject H_0
Precision	1.6785×10^{-3}	Reject H_0
F_1	1.526×10^{-4}	Reject H_0

Table 2: The results of pairwise performance comparisons between multiclass and multilabel models, using a Wilcoxon signed-rank test with the null hypothesis proposing no significant performance difference between the two modelling approaches.

	Class error	Label error	Mean error	Recall	Prec.	F_1	Mean rank
NR	2.6	2.4	2.0	2.8	1.8	2.4	2.33 ± 0.13
CR	1.8	1.8	2.0	1.4	2.0	1.8	1.80 ± 0.18
LR	1.6	1.8	2.0	1.8	2.2	1.8	1.87 ± 0.12

Table 3: The mean relative performance ranks per metric for the **multiclass models** of each resampling scenario (abbreviated), across five evaluation sets. The best performance rank for each metric is emphasised in bold.

	Class error	Label error	Mean error	Recall	Prec.	F_1	Mean Rank
NR	1.8	1.6	1.2	2.6	1.6	1.6	1.73 ± 0.13
CR	2.0	2.0	2.6	2.2	2.0	1.6	2.07 ± 0.15
LR	2.2	2.4	2.2	1.2	2.8	2.4	2.20 ± 0.16

Table 4: The mean relative performance ranks per metric for the **multilabel models** of each resampling scenario (abbreviated), across five evaluation sets. The best performance rank for each metric is emphasised in bold.

This result may also indicate that the resampling strategies applied are too aggressive, and introduced more noise than value.

4.2 Experimental augmentation performance

The analysis of the augmentation experiment results can be divided in two parts. The aggregated performance measures (included in Appendix A, and briefly discussed in Section 4.1) provide overall measures of the models' robustness to augmentation. Additionally, the change in model performance as the intensity of augmentation increases is used to provide a more detailed view of how robust each model is to augmentation, and to which extent a model maintains its performance.

4.2.1 Aggregated experimental performance

The aggregated performance results show that multilabel models prove more robust to brightness and noise augmentations than multiclass models, achieving better performance in nearly all metrics and resampling scenarios. The rotation experiment, however, yielded more varied results, with the multiclass model outperforming the multilabel model under the *class resampled* scenario, while multilabel models produce more robust performance in the other two resampling scenarios.

Resampled models demonstrate a lower deterioration in performance under the simulated low light conditions of the brightness experiment. The *label resampled* models achieve better performance than the other resampling scenarios in all metrics except *precision*. For multilabel models, the *class resampled* scenario yields the best precision, while the *no resampling* scenario only yields the best precision for multiclass models.

Regarding simulated white noise, data resampling yields improved robustness for multiclass models, with the *label resampled* scenario yielding the best performance in four metrics. The *class resampled* and *no resampling* scenarios yield the best multiclass

model in recall and precision respectively. On the other hand, multilabel models proved to be most robust to noise in the *no resampling* scenario, all metrics beside recall, where the *label resampled* scenario performs best.

Lastly, the *class resampled* multiclass models are most robust to sample rotation across metrics, with the exception of high precision obtained by the *no resampling* multiclass model. Similar to the noise experiment, the *no resampling* multilabel models performs best in all metrics except class error and recall, where the *label resampled* scenario demonstrated high performance.

In summary, resampled multiclass models achieve more robust performance under experimental conditions, with the *label* and *class resampled* scenario yielding the best performance in nine and six cases respectively, across experiments and metrics, while *no resampling* multiclass model obtains the highest precision in all three experiments. For multilabel models, however, the resampled scenarios yield more robust performance only in the brightness experiment, while the *no resampling* multilabel models are more robust to noise and rotation. Overall, the *label resampled* scenario yields the most robust multilabel performance in eight cases, the *class resampled* scenario only once (on precision in the brightness experiment), while the *no resampling* multilabel model performs best in a total of nine metrics across the noise and rotation experiments.

4.2.2 Granular experimental performance

Figure 7 shows in more detail the F_1 score performance of each model, as the intensity of brightness reduction increases, plotted as $1 - F_1$ versus γ . In most cases, performance deteriorates notably as the intensity increases. The *class resampled* multiclass model exhibits the fastest deterioration, and both *class* and *label resampled* multiclass models achieve worse performance than the *no resampling* multiclass model, despite achieving good performance at low intensities. The *label resampled* multilabel model, on the other hand, exhibits very stable performance across the intensities investigated, further demonstrating the high robustness observed in the aggregated performance results. Furthermore, across all resampling scenarios, the multilabel models consistently show better performance than multiclass models. Whilst not shown here, it is worth noting that the class and label error, as well as recall performances reveals similar behaviour as intensity increases, while precision performances deviate from this pattern slightly, in line with earlier observations on model precision.

Comparable observations hold regarding model robustness to noise, as highlighted with the class error plotted against the random noise deviation (σ) in Figure 8. As in the aggregated performance, the *label resampled* scenario yields the lowest error for multiclass models throughout, although the *no resampling* scenario does seem to yield more stable performance, obtaining a similar error score at the most extreme intensities tested. For multiclass models all scenarios yield more stable performance than multiclass models, with the *no resampling* scenario maintaining best performance overall. Similar patterns hold for other metrics, with the exception of recall, where the *no resampling* scenario is generally less robust.

The model performance as rotation increases is less conclusive, with the performance of all models tend to deteriorating at similar rates as the rotation angle increases. However, some slight trends do emerge across different metrics. For example, while the *no resampling* models typically show higher error scores initially, they tend to be slightly more stable compared to resampled models, in many cases obtaining the lowest error scores at the most extreme rotation angles. Additionally, while the *no resampling* and *label resampled* multilabel models outperform the multiclass models at

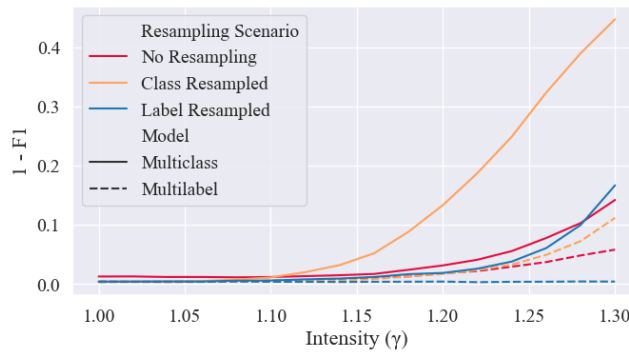


Figure 7: Average $1 - F_1$ performance of each model as the intensity of brightness reduction increases, over 15 trials. A higher gamma-value corresponds to more brightness reduction.

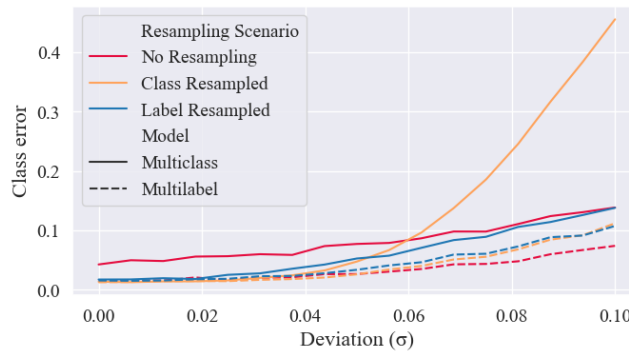


Figure 8: Average class error performance of each model as the level of random noise increases, over 15 trials. The x-axis denotes the standard deviation of the randomly sampled Gaussian noise.

all rotation angles, this result is often reversed in the *class resampled* scenario. There the multiclass models show better class, label and mean errors, as well as recall and F_1 scores at high rotation angles, which is evident as well in the aggregated performance discussed in Section 4.2.1.

These observations are best seen in the mean error metric, where the separate model performances are more easily distinguished. Figure 9 shows the mean error performance plotted against the absolute rotation angle θ . While the multiclass models typically have higher error rates than multilabel models, especially at lower rotation angles, the *class resampled* multilabel model does return a higher mean error than the corresponding multiclass model, when the rotation angle is greater than 20° . Additionally, while the *no resampling* multiclass model has high mean errors overall, it exhibits better performance than the *label resampled* multiclass model, and similar performance to the *class resampled* multiclass model, at high rotations. The *no resampling* multilabel models shows similarly stable performance, further highlighting that models from the two resampled scenarios are not as robust to rotated samples as the corresponding *no resampling* models.

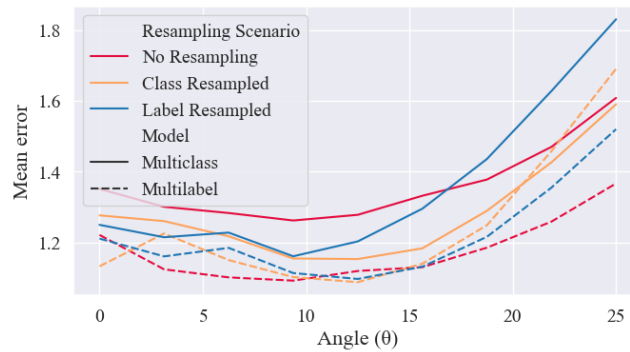


Figure 9: Mean error performance of each model as the angle of rotation increases, averaged over 15 trials. The x-axis denotes the absolute rotation angle, either clockwise or counter-clockwise.

It is worth noting that, across all experiments, the mean error performances of models exhibits the most erratic performance changes as the intensity increases, at times improving briefly, even though the overall trend clearly demonstrates a performance decrease as intensity increases. This could indicate trade-offs between class and label errors — a lower mean error rate could be achieved by reducing the number of misclassifications with multiple dot mislabels, thereby directly reducing the degree of errors, or by increasing the number of misclassifications with single-dot mislabels, effectively reducing the weight of high degree misclassifications. The erratic movement of this mean error over small intensity intervals likely corresponds to small changes in class and label errors that are unnoticed in each individual metric, but interfere either constructively or destructively with each other, yielding larger increases or decreases in the mean error.

4.3 Class imbalance and model performance

Tables A7–A12 in Appendix A reports the mean class performances across each class group, for each of the six models.

Multilabel models consistently have a better middle class baseline performance than multiclass models, and their performance on majority and minority classes typically deviate less from this baseline. This pattern is most evident in the *no resampling* scenario, where class imbalance is most prevalent, further demonstrating that the multilabel model is more robust to imbalance in the training data. Figure 10 shows a granular view of the class error results on each class in the majority and minority groups. Classes are ordered by frequency, such that within each group, the class with the most training samples in the *no resampling* scenario appears first on the x-axis.

The multiclass and multilabel models show similar performance in the *class resampled* scenario, while the multiclass model exhibits a high error on a number of classes in the *label resampled* scenario. It was noted that, in this scenario, four of the majority classes were undersampled to such a degree that they were among the least frequent classes in the resampled training set, which could explain the relatively poor performance of the multiclass model on some of the majority classes. The performance

deterioration of the multiclass model under the *no resampling* scenario is evident, including four minority classes with a classification error rate of 100%.



Figure 10: Class error performance of different models on each class in the majority and minority groups, along with a rolling mean (window size 5) for each group. The average middle class performance is included as baseline. Classes are ordered by training sample frequency within each class group.

In both the *class* and *label resampled* scenarios, a few cases are observed where the multiclass and multilabel models have a comparable deviation from their respective baselines, especially in majority class performance. However, in most of these cases, multilabel models still yield better relative performance on minority classes. Furthermore, a number of instances are noted where the multilabel performance on minority classes is comparable to, or better than, the corresponding multiclass performance on middle classes, further supporting the conclusion that multilabel models are not as sensitive to imbalanced data as multiclass models. This includes the class and label errors under the *class resampled* scenario, the recall under all scenarios. It is especially noted that the multilabel models achieve lower mean error scores on minority classes than the multiclass models do on middle, majority and minority classes.

Across models, the *no resampling* scenario typically yields the highest performance deviation on majority and minority classes, while the *class resampled* scenario yields the lowest deviation, outperforming the *label resampled* scenario in nearly all cases. This observation holds for both multiclass and multilabel models, indicating that both model approaches do benefit from data resampling, in different degrees.

In both the *class* and *label resampled* scenarios, models yield a higher mean error performance on majority and minority classes compared to their baselines. This is contrary to the expected improvement in performance on majority classes, and deterioration in performance on minority classes. However, the *no resampling* scenario shows improved performance on minority and majority classes, or only slightly worse performance on minority classes for the multilabel model, while still yielding a higher baseline performance compared to other resampling scenarios. This indicates that, while the *no resampling* scenario yields more classification errors overall, indicated by class and label

error metrics, this predominantly includes small degree errors such as mislabelling single dot positions. Models trained in the *class* and *label resampled* scenarios, yield fewer misclassifications, and in particular make fewer small degree errors, leading to large mean error rates.

Figure 11 shows a granular view of the mean error results on majority and minority classes. Multiclass and multilabel models again show similar performance in the *class resampled* scenario, while the *no resampling* and *label resampled* multiclass models yield worse performance on minority and majority classes compared to their respective baselines.



Figure 11: Mean error performance of different models on each class in the majority and minority groups, along with a rolling mean (window size 5) for each group. The average middle class performance is included as baseline. Classes are ordered by training sample frequency within each class group.

5 Conclusion

This study shows that the use of a multilabel classification framework, combined with data resampling, yields a high performing model able to generalise to unseen data, including different Braille language codes with unseen Braille character classes and distinct class and label distributions. This approach emphasises a focus on Braille dots, rather than Braille characters, reducing the complexity of the target labels and reducing the impact of language bias, or class imbalance. This bias is further mitigated by resampling, reducing the prevalence of frequently occurring label patterns and simulating valid instances of less frequent, but equally important Braille samples.

Multilabel models outperform multiclass models across scenarios, including on test data from different languages. This is emphasised by the stable performance of multilabel models on majority and minority classes, as well as under poor data quality conditions. When tested on out-of-distribution samples of character classes with little to no training

data, multilabel models are able to achieve reasonable performance, while multiclass models are unable to identify these characters.

Resampling to account for data imbalance improves modelling result under some conditions, especially for multiclass models. The experimental results indicate that multiclass models trained on resampled data are more robust to adverse quality conditions. For the multilabel approach, while resampling yielded models more robust to lower brightness, these models are less robust to noise and rotation. This decrease in model robustness may relate to an increase in ambiguity between recto- and verso-dots in the synthetic oversampled training samples. Furthermore, in some cases the data resampling strategies used includes extreme undersampling of majority character classes, leading to a decrease in performance on these classes.

Further study of resampling strategies for Braille recognition could yield more optimised and robust models. This optimal resampling strategy could depend on multiple factors, including model design, training data distributions, the expected or most prevalent data quality issues, and the required balance between over- and undersampling. Furthermore, the use of the multilabel classification framework could be extended to different CNN architectures, and combined with different convolutional Braille recognition techniques already explored in literature.

References

- [Al-Salman and Al-Salman, 2024] Al-Salman, A.-M. and Al-Salman, A. (2024). Fly-LeNet: A deep learning-based framework for converting multilingual Braille images. *Heliyon*, 10(4):26155.
- [Al-Salman et al., 2007] Al-Salman, A.-M. S., Al-Ohali, Y., Al-Abdulkarim, L. O., and Salman, Y. (2007). Trends and technologies in optical Braille recognition. In *Proceedings of the 3rd International Conference on Information Technology*, Amman.
- [Alufaisan et al., 2021] Alufaisan, S., Albur, W., Alsedrah, S., and Latif, G. (2021). Arabic Braille numeral recognition using convolutional neural networks. In Bindhu, V., Tavares, J. M. R. S., Boulogeorgos, A.-A. A., and Vuppapapati, C., editors, *Proceedings of the International Conference on Communication, Computing and Electronics Systems*, pages 87–101, Singapore. Springer Singapore.
- [Antonacopoulos and Bridson, 2004] Antonacopoulos, A. and Bridson, D. (2004). A robust braille recognition system. In Marinai, S. and Dengel, A., editors, *Proceedings of the 6th International Workshop on Document analysis systems VI*, volume 3163 of *Lecture Notes in Computer Science*, pages 533–545.
- [Balaprakash et al., 2007] Balaprakash, P., Birattari, M., and Stützle, T. (2007). Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In Bartz-Beielstein, T., Blesa Aguilera, M. J., Blum, C., Naujoks, B., Roli, A., Rudolph, G., and Sampels, M., editors, *Hybrid Metaheuristics*, pages 108–122, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [Baumgärtner et al., 2020] Baumgärtner, C., Schwarz, T., and Stiefelhagen, R. (2020). Image-based recognition of Braille using neural networks on mobile devices. In Miesenberger, K., Manduchi, R., Covarrubias Rodriguez, M., and Peñáz, P., editors, *Computers Helping People with Special Needs*, pages 346–353, Cham. Springer International Publishing.
- [Bipin Nair et al., 2025] Bipin Nair, B., Niranjan, Saketh, P., and Rani, N. S. (2025). Wayvision: A hybrid deep learning approach for recognizing handwritten Kannada Braille using wavelet transformation and attention based YOLOv5. *MethodsX*, page 103440.
- [Birattari et al., 2002] Birattari, M., Stützle, T., Paquete, L., and Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18, San Francisco.

- [Braille Character dataset, 2019] Braille Character dataset (2019). Available <https://www.kaggle.com/datasets/shanks0465/Braille-character-dataset>.
- [Calders et al., 1986] Calderys, P., Mennens, J. E., and Francois, G. E. (1986). Optical pattern recognition of Braille originals. In *Proceedings of the SPIE 0655, Optical System Design, Analysis, Production for Advanced Technology Systems*, volume 0655, Innsbruck.
- [Devi and Baboo, 2012] Devi, A. and Baboo, S. (2012). Computer based Tamil Braille system – a review. In *Proceedings of the 11th International Tamil Internet Conference*, pages 57–61, Chidambaram.
- [Elaraby et al., 2024] Elaraby, N., Barakat, S., and Rezk, A. (2024). A generalized ensemble approach based on transfer learning for Braille character recognition. *Information Processing and Management*, 61(1).
- [Gezahegn et al., 2019] Gezahegn, H., Su, T.-Y., Su, W.-C., and Ito, M. (2019). An optical Braille recognition system for enhancing Braille literacy and communication between the blind and non-blind. *Review of Undergraduate Computer Science*, 2018(2):1–5.
- [Gonçalves et al., 2020] Gonçalves, D., dos Santos, G., de Borba Campos, M., Amory, A., and Manssour, I. (2020). Braille character detection using deep neural networks for an educational robot for visually impaired people. In *Proceedings of the XVI Workshop of Computer Vision*, pages 123–128.
- [Hanumanthappa and Murthy, 2016] Hanumanthappa, M. and Murthy, V. V. (2016). Optical Braille recognition and its correspondence in the conversion of Braille script to text — a literature review. In *Proceedings of the International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, pages 297–301.
- [He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.
- [Howard et al., 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861.
- [Hsu, 2020] Hsu, B.-M. (2020). Braille recognition for reducing asymmetric communication between the blind and non-blind. *Symmetry*, 12(1069):1–15.
- [Huang et al., 2017] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269.
- [Iandola et al., 2016] Iandola, F., Han, S., Moskewicz, M., Ashraf, K., Dally, W., and Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5mb model size. *ArXiv*.
- [Isayed and Tahboub, 2015] Isayed, S. and Tahboub, R. (2015). A review of optical Braille recognition. In *Proceedings of the 2nd World Symposium on Web Applications and Networking (WSWAN)*, pages 1–6.
- [Jocher et al., 2022] Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., NanoCode012, Kwon, Y., Michael, K., TaoXie, Fang, J., imyhxy, Lorna, Yifu, Z., Wong, C., V, A., Montes, D., Wang, Z., Fati, C., Nadar, J., Laughing, UnglvKitDe, Sonck, V., tkianai, yxNONG, Skalski, P., Hogan, A., Nair, D., Strobel, M., and Jain, M. (2022). Ultralytics YOLOv5: v7.0 - YOLOv5 SOTA realtime instance segmentation. Published as software package.
- [Kaur et al., 2020] Kaur, P., Ramu, S., Panchakshari, S., and Krupa, N. (2020). Conversion of Hindi Braille to speech using image and speech processing. In *Proceedings of the IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, volume 7, pages 1–6.

- [Kausar et al., 2021] Kausar, T., Manzoor, S., Kausar, A., Lu, Y., Wasif, M., and Ashraf, M. A. (2021). Deep learning strategy for Braille character recognition. *Institute of Electrical and Electronics Engineers Access*, 9:169357–169371.
- [Ke et al., 2017] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- [Kingma and Ba, 2014] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.
- [Klazar and Engelbrecht, 2014] Klazar, R. and Engelbrecht, A. P. (2014). Parameter optimization by means of statistical quality guides in F-Race. In *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2547–2552.
- [Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.
- [Kumar and Basha, 2025] Kumar, T. S. and Basha, S. M. (2025). Braille character recognition for blind people using mobilenet compared with LSTM model. *Applications of Mathematics in Science and Technology: International Conference on Mathematical Applications in Science and Technology*, pages 729–733.
- [Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Institute of Electrical and Electronics Engineers*, 86(11):2278–2324.
- [Li and Yan, 2021] Li, C. and Yan, W. (2021). Braille recognition using deep learning. In *Proceedings of the 4th International Conference on Control and Computer Vision, ICCCV '21*, pages 30–35.
- [Li et al., 2018] Li, R., Liu, H., Wang, X., and Qian, Y. (2018). DSBI: double-sided Braille image dataset and algorithm evaluation for Braille dots detection. In *Proceedings of the 2nd International Conference on Video and Image Processing, ICVIP '18*, pages 65–69. Association for Computing Machinery.
- [Li et al., 2020] Li, R., Liu, H., Wang, X., Xu, J., and Qian, Y. (2020). Optical Braille recognition based on semantic segmentation network with auxiliary learning strategy. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2362–2368.
- [Lin et al., 2017] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007.
- [Liu et al., 2009] Liu, X.-Y., Wu, J., and Zhou, Z.-H. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550.
- [Meng et al., 2024] Meng, Z., Cai, Z., Feng, J., Ma, H., Zhang, H., and Li, S. (2024). Braille character segmentation algorithm based on gaussian diffusion. *Computers, Materials and Continua*, 79(1):1481–1496.
- [Mennens et al., 1994] Mennens, J., Van Tichelen, L., Francois, G., and Engelen, J. (1994). Optical recognition of Braille writing using standard equipment. *Proceedings of the IEEE Transactions on Rehabilitation Engineering*, 2(4):207–212.
- [Morgavi and Mauro, 2002] Morgavi, G. and Mauro, M. (2002). A neural network hybrid model for an optical Braille recognizer. *International Conference on Signal, Speech and Image Processing*.
- [Murthy and Hanumanthappa, 2022] Murthy, V. V. and Hanumanthappa, M. (2022). Vgg16 cnn based Braille cell classifier model for translation of Braille to text. *Specialusis Ugdymas*, 1(43):4388–4396.

- [Ovodov, 2021a] Ovodov, I. G. (2021a). Optical Braille recognition using object detection CNN. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 1741–1748.
- [Ovodov, 2021b] Ovodov, I. G. (2021b). Semantic-based annotation enhancement algorithm for semi-supervised machine learning efficiency improvement applied to optical Braille recognition. In Proceedings of the IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), pages 2190–2194.
- [Redmon and Farhadi, 2017] Redmon, J. and Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6517–6525.
- [Revelli et al., 2022] Revelli, V. P., Sharma, G., and Kiruthika Devi, S. (2022). Automate extraction of Braille text to speech from an image. *Advances in Engineering Software*, 172(3):103180–103185.
- [Shao et al., 2022] Shao, Z., Yu, Z., Gu, Y., and Wang, W. (2022). Braille-to-Chinese translation system based on optical Braille recognition. In Proceedings of the Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS), volume 2, pages 22–26.
- [Shapiro and Wilk, 1965] Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3–4):591–611.
- [Shimomura et al., 2018] Shimomura, Y., Kawabe, H., Nambo, H., and Seto, S. (2018). Construction of restoration system for old books written in Braille. In Proceedings of the International Conference on Management Science and Engineering Management, pages 469–477.
- [Shokat et al., 2020] Shokat, S., Riaz, R., Rizvi, S. S., Abbasi, A. M., Abbasi, A. A., and Kwon, S. J. (2020). Deep learning scheme for character prediction with position-free touch screen-based Braille input method. *Human-centric Computing and Information Sciences*, 10(41):1–24.
- [Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.
- [Sivasamy et al., 2013] Sivasamy, P., S. M., Reddy, S., and Meenakshy, D. (2013). Conversion of Braille to text in English, Hindi and Tamil languages. *International Journal of Computer Science, Engineering and Applications*, 3.
- [Szegedy et al., 2015a] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015a). Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–9, Los Alamitos, CA, USA. IEEE Computer Society.
- [Szegedy et al., 2015b] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015b). Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–9.
- [Wilcoxon, 1945] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

A Supplementary tables

Tables A1–A6 provide the overall performance measures of the six models on the five evaluation sets. In each table, the pairwise comparison between modelling approaches is emphasised by highlighting the performance of the best model within each resampling scenario, in bold. Furthermore, to aid in comparing different resampling strategies for a given modelling approach, the performance of the best multiclass and multilabel models in each evaluation, are underlined.

Tables A7–A12 report the mean class performances for each of the six models, across three character class groups — minority, majority, and middle classes. For each resampling scenario, the model with the lowest performance deviation in majority and minority class groups is highlighted in bold. For each modelling approach, the resampling scenario with the most stable performance is underlined.

Evaluation set	No resampling		Class resampled		Label resampled	
	MC	ML	MC	ML	MC	ML
ID	0.0474	0.0160	0.0136	<u>0.0137</u>	0.0183	0.0162
OOD	0.2792	0.0611	<u>0.0556</u>	0.0237	0.0891	0.0656
Brightness	0.1058	0.0598	0.3041	0.0740	<u>0.1012</u>	0.0165
Noise	0.0817	0.0340	0.1226	0.0405	<u>0.0613</u>	0.0446
Rotation	0.2743	0.2339	0.2285	0.2591	0.2716	0.2299

Table A1: Class error across three resampling scenarios for multiclass (MC) and multilabel (ML) models. The null hypothesis that multiclass and multilabel models yield similar performance is rejected with a p -value of 2.136×10^{-4} .

Evaluation set	No resampling		Class resampled		Label resampled	
	MC	ML	MC	ML	MC	ML
ID	0.0104	0.0030	<u>0.0027</u>	0.0026	0.0038	0.0032
OOD	0.0502	0.0109	<u>0.0102</u>	0.0043	0.0191	0.0127
Brightness	0.0282	0.0140	0.0913	0.0199	<u>0.0267</u>	0.0033
Noise	0.0196	0.0069	0.0380	0.0090	<u>0.0141</u>	0.0099
Rotation	0.0656	0.0484	0.0538	0.0623	0.0710	0.0513

Table A2: Label error across three resampling scenarios for multiclass (MC) and multilabel (ML) models. The null hypothesis that multiclass and multilabel models yield similar performance is rejected with a p -value of 9.16×10^{-5} .

Evaluation set	No resampling		Class resampled		Label resampled	
	MC	ML	MC	ML	MC	ML
ID	1.3163	1.1302	<u>1.1951</u>	1.1394	1.2455	1.1744
OOD	<u>1.0797</u>	1.0732	1.0958	1.0887	1.2863	1.1622
Brightness	1.5989	1.4045	1.8016	1.6159	<u>1.5797</u>	1.1849
Noise	1.4404	1.2160	1.8587	1.3405	<u>1.3842</u>	1.3340
Rotation	1.4351	1.2414	1.4136	1.4438	1.5698	1.3390

Table A3: Mean error across three resampling scenarios for multiclass (MC) and multilabel (ML) models. The null hypothesis that multiclass and multilabel models yield similar performance is rejected with a p -value of 3.05×10^{-5} .

Evaluation set	No resampling		Class resampled		Label resampled	
	MC	ML	MC	ML	MC	ML
ID	0.0155	0.0049	0.0026	0.0030	0.0037	0.0019
OOD	0.0367	0.0045	<u>0.0011</u>	0.0005	0.0112	0.0011
Brightness	0.0494	0.0102	0.1098	0.0387	<u>0.0307</u>	0.0019
Noise	0.0298	0.0079	<u>0.0076</u>	0.0035	0.0092	0.0024
Rotation	0.1024	0.0784	0.0757	0.1064	0.0966	0.0667

Table A4: 1– recall across three resampling scenarios for multiclass (MC) and multilabel (ML) models. The null hypothesis that multiclass and multilabel models yield similar performance is rejected with a p -value of 1.0071×10^{-3} .

Evaluation set	No resampling		Class resampled		Label resampled	
	MC	ML	MC	ML	MC	ML
ID	0.0056	0.0012	<u>0.0029</u>	0.0023	0.0040	0.0045
OOD	0.0731	0.0197	<u>0.0213</u>	0.0091	0.0310	0.0267
Brightness	0.0080	0.0180	<u>0.0778</u>	0.0018	0.0234	0.0047
Noise	<u>0.0101</u>	0.0060	0.0653	0.0146	0.0192	0.0175
Rotation	<u>0.0328</u>	0.0209	0.0348	0.0217	0.0498	0.0384

Table A5: 1– precision across three resampling scenarios for multiclass (MC) and multilabel (ML) models. The null hypothesis that multiclass and multilabel models yield similar performance is rejected with a p -value of 1.6785×10^{-3} .

Evaluation set	No resampling		Class resampled		Label resampled	
	MC	ML	MC	ML	MC	ML
ID	0.0106	0.0031	<u>0.0028</u>	0.0026	0.0038	0.0032
OOD	0.0552	0.0122	<u>0.0113</u>	0.0048	0.0212	0.0140
Brightness	0.0291	0.0141	0.0941	0.0206	<u>0.0271</u>	0.0033
Noise	0.0201	0.0070	0.0373	0.0091	<u>0.0142</u>	0.0100
Rotation	0.0689	0.0505	0.0557	0.0660	0.0738	0.0527

Table A6: 1– F_1 across three resampling scenarios for multiclass (MC) and multilabel (ML) models. The null hypothesis that multiclass and multilabel models yield similar performance is rejected with a p -value of 1.526×10^{-4} .

Resampling scenario	Multiclass			Multilabel		
	Middle	Majority	Minority	Middle	Majority	Minority
No resampling	0.1271	0.0325 -74.4%	0.6794 +434.6%	0.0570	0.0220 -61.5%	0.1315 +130.6%
Class resampled	0.0490	0.0275 -43.9%	0.0749 +53.1%	0.0296	0.0159 -46.3%	0.0287 -3.0%
Label resampled	0.0569	0.0384 <u>-32.5%</u>	0.1672 +194.0%	0.0497	0.0344 <u>-30.7%</u>	0.1023 +106.0%

Table A7: Class error performance of majority and minority classes compared to the middle class, across three resampling scenarios for multiclass and multilabel models.

Resampling scenario	Multiclass			Multilabel		
	Middle	Majority	Minority	Middle	Majority	Minority
No resampling	0.0244	0.0062 -74.4%	0.1239 +407.5%	0.0099	0.0038 -62.0%	0.0237 +138.9%
Class resampled	0.0087	0.0050 -42.1%	0.0137 +58.5%	0.0050	0.0028 -44.2%	0.0053 +4.8%
Label resampled	0.0102	0.0074 <u>-27.7%</u>	0.0355 +247.0%	0.0087	0.0061 <u>-29.7%</u>	0.0203 +132.4%

Table A8: Label error performance of majority and minority classes compared to the middle class, across three resampling scenarios for multiclass and multilabel models.

Resampling scenario	Multiclass			Multilabel		
	Middle	Majority	Minority	Middle	Majority	Minority
No resampling	1.2308	1.1651 -5.3%	1.1102 -9.8%	1.0560	1.0493 -0.6%	1.0590 +0.3%
Class resampled	1.1020	1.1029 +0.1%	1.1064 +0.4%	1.0310	1.0492 +1.8%	1.0876 +5.5%
Label resampled	1.1619	1.2574 +8.2%	1.3018 +12.0%	1.0556	1.0573 +0.2%	1.1357 +7.6%

Table A9: Mean error performance of majority and minority classes compared to the middle class, across three resampling scenarios for multiclass and multilabel models.

Resampling scenario	Multiclass			Multilabel		
	Middle	Majority	Minority	Middle	Majority	Minority
No resampling	0.0352	0.0063 -82.0%	0.0873 +148.0%	0.0051	0.0023 -55.7%	0.0081 +58.5%
Class resampled	0.0012	0.0005 -55.1%	0.0019 +64.7%	0.0006	0.0003 -59.8%	0.0007 +5.5%
Label resampled	0.0074	0.0059 <u>-21.0%</u>	0.0202 +172.0%	0.0012	0.0008 <u>-35.8%</u>	0.0012 +4.9%

Table A10: 1– recall performance of majority and minority classes compared to the middle class, across three resampling scenarios for multiclass and multilabel models.

Resampling scenario	Multiclass			Multilabel		
	Middle	Majority	Minority	Middle	Majority	Minority
No resampling	0.0115	0.0112 -2.5%	0.1668 +1353.1%	0.0146	0.0096 -34.3%	0.0694 +376.3%
Class resampled	0.0184	0.0142 -22.5%	0.0402 +119.3%	0.0110	0.0095 -14.1%	0.0209 +89.2%
Label resampled	0.0158	0.0126 -20.4%	0.0780 +394.0%	0.0190	0.0179 -5.8%	0.0695 +266.7%

Table A11: 1– precision performance of majority and minority classes compared to the middle class, across three resampling scenarios for multiclass and multilabel models.

Resampling scenario	Multiclass			Multilabel		
	Middle	Majority	Minority	Middle	Majority	Minority
No resampling	0.0246	0.0090 -63.3%	0.1451 +490.8%	0.0103	0.0060 -41.3%	0.0429 +317.6%
Class resampled	0.0102	0.0075 -26.6%	0.0222 +117.6%	0.0062	0.0049 -20.6%	0.0111 +78.8%
Label resampled	0.0121	0.0093 -23.2%	0.0525 +334.6%	0.0105	0.0095 -9.5%	0.0390 +269.8%

Table A12: 1– F_1 performance of majority and minority classes compared to the middle class, across three resampling scenarios for multiclass and multilabel models.